МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
имени М.В.Ломоносова

Факультет вычислительной математики и кибернетики

Практическое задание № 2 по учебному курсу «Численные методы линейной алгебры» Метод Чебышева с оптимальным набором итерационных параметров Матрица №2

ОТЧЁТ

о выполненном задании

студента **301** учебной группы факультета ВМК МГУ

Матвеевой Софьи Вячеславовны

гор. Москва

2023г.

Постановка задачи

Требуется методом Чебышева с оптимальным набором итерационных параметров приближенно решить систему линейных алгебраических уравнений

$$x + Ax = F$$

с симметричной положительно определенной матрицей $A \in R^{n\times n}$. Элементы матрицы a_{ij} являются вещественными числами, расположенными на отрезке $[-1;\ 1]$(за исклбчением диогональных элементов). Матрица предоставляется в виде файла в формате csv.

С помощью теоремы Гершгорина оценить спектр матрицы системы уравнений. Подобрать наименьший показатель степени, при котором погрешность решения на последней итерации в среднеквадратической норме не превосходит погрешность прямого метода(метода Холецкого). Построить график среднеквадратической нормы погрешности решения как функции номера итерации метода Чебышева.

Для проверки метода требуется случайно сгенерировать вектор-столбец $\boldsymbol{x}$ с равномерно распределенными на отрезке [-1; 1] компонентами, вычислить правую часть по формуле $F = Ax$.

Описание метода решения задачи :

1. Оценка спектра с помощью теоремы Гершгорина.

Теорема Гершгорина: Каждое характеристическое число матрицы А всегда расположено в одном из кругов:

$$|\boldsymbol{a_{ii}} - \boldsymbol{\lambda}| \le \sum_{\boldsymbol{j \ne i, j = 1}}^{\boldsymbol{n}} |\boldsymbol{a_{ij}}| \qquad \boldsymbol{i = 1, \ldots. n}$$

Так как матрица вещественна и является матрицей с диагональным преобладанием , то все собственные числа являются вещественными. Тогда для оценки спектра матрицы А найдем:

$$h_1 = \min_i(a_{ii} - \sum_{\boldsymbol{j \ne i, j = 1}}^{\boldsymbol{n}} |\boldsymbol{a_{ij}}|), \quad \boldsymbol{i = 1, \ldots.. n}$$

$$h_2 = \max_i(a_{ii} + \sum_{\boldsymbol{j \ne i, j = 1}}^{\boldsymbol{n}} |\boldsymbol{a_{ij}}|), \quad \boldsymbol{i = 1, \ldots.. n}$$

2. Решение СЛАУ с помощью итерационного метода Чебышева

Обозначим А = Ax + x.

Для решения Ax = f методом Чебышева проведём последовательные приближения вектора x. По заданию $x^0 = (0,......,0)^T$, (n+1)-е приближение найдём по формуле:

$$\frac{x^{n+1} - x^n}{\tau_{n+1}} + Ax^n = f, \quad n = 0,....m$$

Выразим:

$$x^{n+1} = x^n - \tau_{n+1}(Ax^n - f), \quad n = 0,....m,$$

Где m – число итераций,

$$\tau_n = \frac{\tau_0}{1 - \rho_0 * cos(\frac{\pi}{2m} * J_k^m)}$$

$$\tau_0 = \frac{2}{h_1 + h_2}$$

$$\rho_0 = \frac{h_2 - h_1}{h_2 + h_1}$$

$h_2, h_1 -$ границы спектра

J_k^m- *итерационный параметр, который находится следующим образом:*

$$J^1 = \{1\}$$

Далее для m являющихся степенью двойки:

$$J_{2p-1}^m = J_p^{m/2}, \quad p = 1,....m/2$$

$$J_{2p}^m = 4p - J_p^{m/2}, \quad p = 1,....m/2$$

Листинг программы

Программа написана на языке «C++14».

В программе были использованы следующие библиотеки и using:

```
#include <iostream>
#include <vector>
#include <string>
#include <fstream>
#include <random>
#include <chrono>
#include <cmath>
#include <fstream>

using namespace std;
using namespace chrono;
```

В программе используются следующие псевдонимы для типов данных:

```
typedef vector<vector<double>> matrix;
typedef vector<double> vec;
```

matrix – тип матриц вещественных чисел
vec – тип векторов вещественных чисел

Функция, которая производит умножение матрицы на вектор-столбец:

```
vec MulMV(const matrix &m, const vec &x)
{
    vec res = vec(m.size(), 0.);
    for (size_t i = 0; i < m.size(); i++)
    {
        for (size_t j = 0; j < m.size(); j++)
        {
            res[i] += m[i][j] * x[j];
        }
    }
    return res;
}
```

Оператор разности двух векторов:

```
vec operator-(const vec &x, const vec &y)
{
    vec ans(x.size(), 0);
    for (size_t i = 0; i < x.size(); i++)
    {
        ans[i] = x[i] - y[i];
    }
    return ans;
}
```

Оператор суммы двух векторов:

```
vec operator+(const vec &x, const vec &y)
{
    vec ans(x.size(), 0);
    for (size_t i = 0; i < x.size(); i++)
    {
        ans[i] = x[i] + y[i];
    }
    return ans;
}
```

Оператор умножения вектора на число:

```
vec operator*(const vec &x, double &y)
{
    vec ans(x.size(), 0);
    for (size_t i = 0; i < x.size(); i++)
    {
        ans[i] = x[i] * y;
    }
    return ans;
}
```

Класс системы линейных алгебраических уравнений:

```
struct Equasions
{
    long N = 0; //Размерность матрицы
    matrix Matrix; //Матрица
    vec x; //Правильный ответ
    vec res; //Полученный ответ
    vec f; //Соответствущая x правая часть
    double h1, h2;//ограничение на спектор матрицы
    vector<uint64_t> it_par = {1};//вектор итерационных параметров
    vec error;//вектор среднеквадратической нормы погрешности
```

Конструктор класса: считывает из заданного файла матрицу, считает размерность матрицы:

```
Equasions(string filename)
    {
        string line;
        ifstream in(filename);
        if (in.is_open())
        {
            while (getline(in, line))
            {
                if (line.empty())
                    break;
                Matrix.emplace_back();
                for (size_t i = 0; i < line.length(); i++)
                {
                    string x = "";
                    while (line[i] != ',' && i != line.length())
                    {
                        x += line[i];
                        ++i;
                    }
                    Matrix[N].push_back(stod(x, nullptr));
                }
                ++N;
            }
        }
        in.close();
    }
```

Функция, которая случайным образом генерирует вектор-столбец решений x с равномерно распределёнными на отрезке [-1; 1] компонентами:

```
void Equasions::CalculateRandX(const unsigned int seed)
  {
    x = vec(N, 0);

    default_random_engine rand(seed);
    uniform_real_distribution<double> dist(0.0, 1.0);

    for (size_t i = 0; i < N; i++)
    {
      x[i] = dist(rand);
    }
  }
```

Функция, вычисляющая правую часть системы уравнений по вектору x:

```
  void Equasions::CalculateF()
  {
    f = MulMV(Matrix, x);
  }
```

Функция, вычисляющая среднеквадратическую норму погрешности решения:

```
  double RMSE()
  {
    double ans = 0;
    for (int i = 0; i < N; ++i)
    {
      ans += (res[i] - x[i]) * (res[i] - x[i]);
    }
    return sqrt(ans / N);
  }
```

Функция, вычисляющая относительную погрешность решения:

```
  double Relative_Error()
  {
    double ans = 0;
    double nx = 0;
    for (int i = 0; i < N; ++i)
    {
      ans += (res[i] - x[i]) * (res[i] - x[i]);
      nx += x[i] * x[i];
    }
    return sqrt(ans) / sqrt(nx);
  }
```

Функция, оценивающая спектр матрицы:

```
void Interval_Characteristics()
  {
    double sum = 0;
    double min = numeric_limits<double>::max();
    double max = 0;
    for (size_t i = 0; i < N; i++)
    {
      double center = fabs(Matrix[i][i]);
      sum = 0;
      for (size_t j = 0; j < N; j++)
      {
        sum += fabs(Matrix[i][j]);
      }
      sum -= center;
      if (center - sum < min)
      {
        min = center - sum ;
      }
      if (center + sum > max)
      {
        max = center + sum ;
      }
    }
    h1 = min;
    h2 = max;
  }
```

Функция, которая находит итерационные параметры:

```
void Iteration_Parameters()
  {
    vector<uint64_t> new_par;
    uint64_t m = it_par.size();
    for (size_t i = 0; i < it_par.size(); i++)
    {
      new_par.push_back(it_par[i]);
      uint64_t new_p = 4 * m - it_par[i];
      new_par.push_back(new_p);
    }
    it_par = new_par;
  }
```

Функция, котрая реализует метод Чебышёва:

```
  void Chebyshev_method()
  {
    double tau0 = 2 / (h1 + h2);
    double r0 = (h2 - h1) / (h2 + h1);
    res = vec(N, 0);
    uint64_t m = it_par.size();
    error = vec(m, 0);
    for (size_t i = 0; i < m; i++)
```

```
    {
        double tauk = tau0 / (1 - r0 * cos(M_PI * it_par[i]/ (2 * m)));
        res = (f - MulMV(Matrix, res)) * tauk + res;
        error[i] = RMSE();
    }
}
```

Main:

```
int main()
{
    Equasions m = Equasions("SLAU_var_2.csv"); //Считывает матрицу из файла
    for (size_t i = 0; i < m.N; i++)//Ax+x=f <=> (A+I)x=f
    {
        m.Matrix[i][i] += 1;
    }

    if (!CheckMatrix(m)) //Проверка на симметричность
    {
        cout << "Матрица не является симметричной" << endl;
        return 0;
    }

    Equasions copy_m = m;// Сохраняет копию матрицы
    double norm = 0; //Переменная для подсчета средней среднеквадратической нормы погрешности
    for (size_t i = 0; i < 100; i++) //Производим 100 итераций вычисления решения системы уравнений
    {
        m.CalculateRandX(i);
        m.CalculateF();m.CalculateL();

        m.FirstGaussBackward();
        m.SecondGaussBackward();
        norm += m.RMSE(); //Считаем среднеквадратическую погрешность
        m = copy_m; //возвращаем матрице ее первоначальные значения
    }

    double Kholetsky_error = norm / 100;
    cout << "Среднеквадратическую погрешность методом Холецкого = " << Kholetsky_error << endl;

    m.Interval_Characteristics();
    cout << "Оценки спектра: h1 = " << m.h1 << " h2 = " << m.h2 << endl;

    copy_m = m;// Сохраняет копию матрицы
    double Chebyshev_error = numeric_limits<double>::max();
    int n = -1;//степень двойки количества итераций
    while(Chebyshev_error > Kholetsky_error)
    {
        ++n;
        Chebyshev_error = 0;
        for (size_t i = 0; i < 10; i++)
        {
            m.CalculateRandX(i);
```

```
        m.CalculateF();
        m.Chebyshev_method();
        Chebyshev_error += m.RMSE();
      }
      Chebyshev_error /= 10;
      cout << "Среднеквадратической отклонение при числе итераций " << (1 << n) << " = "<< Chebyshev_error << endl;
      m.Iteration_Parameters();
    }

  cout << "Наименьший показатель степени двойки = " << n << endl;
  cout << "Относительная погрешность решения, полученного методом Чебышева = " << m.Relative_Error() << endl;
  ofstream file;
  file.open("Errors.txt");
  file << '[';
  for (auto elem : m.error) {
    file << std::fixed << std::setprecision(20) << elem << ", ";
  }
  file << ']';
  file.close();
  return 0;
}
```

Полученные результаты

Среднеквадратическую погрешность методом Холецкого = 2.19438e-16

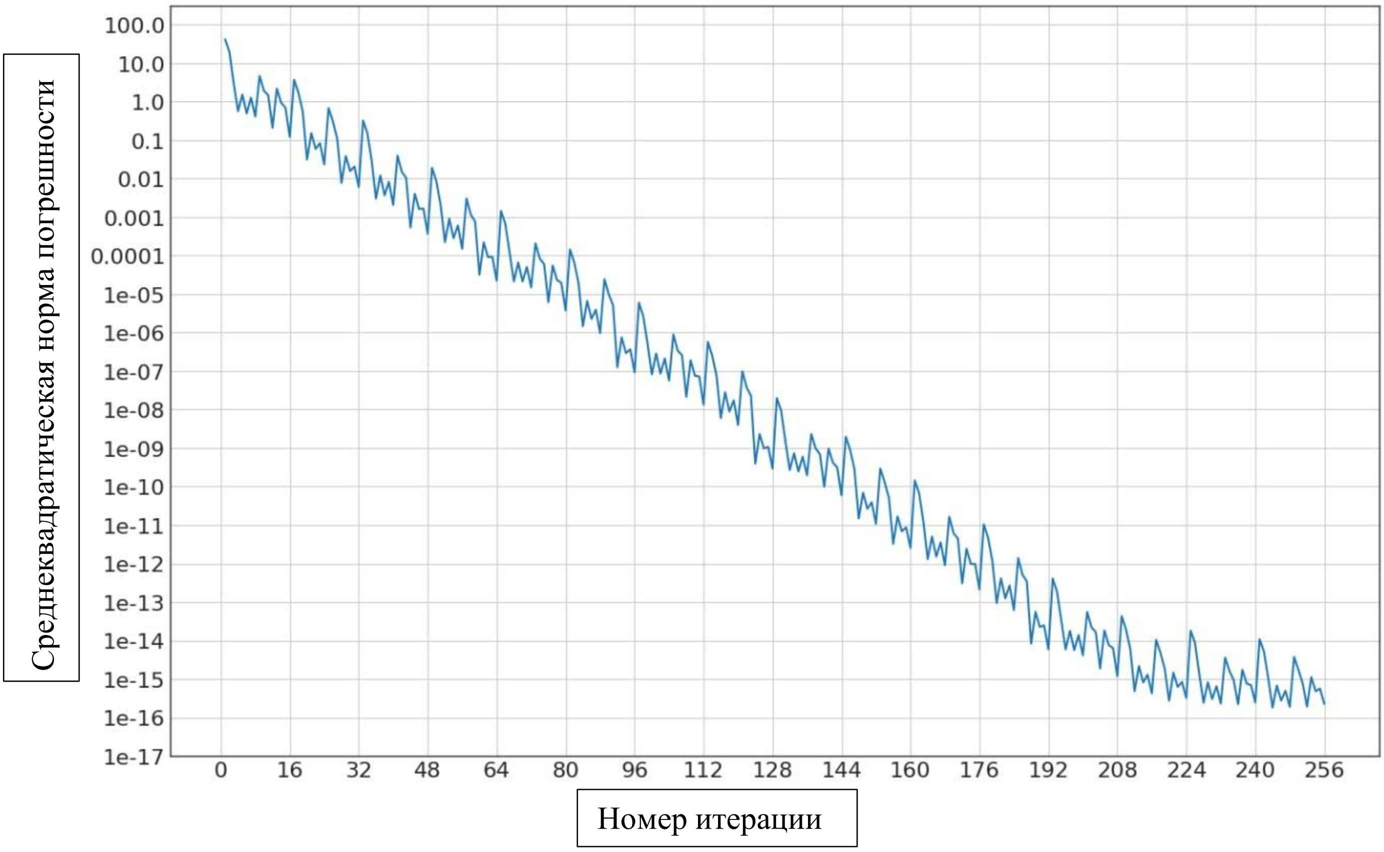
Оценки спектра: h1 = 1 h2 = 153.4

Среднеквадратической отклонение при числе итераций 1 = 0.0920647
Среднеквадратической отклонение при числе итераций 2 = 0.531929
Среднеквадратической отклонение при числе итераций 4 = 0.407347
Среднеквадратической отклонение при числе итераций 8 = 0.223949
Среднеквадратической отклонение при числе итераций 16 = 0.0615709
Среднеквадратической отклонение при числе итераций 32 = 0.00501245
Среднеквадратической отклонение при числе итераций 64 = 2.58137e-05
Среднеквадратической отклонение при числе итераций 128 = 8.01435e-10
Среднеквадратической отклонение при числе итераций **256** = **1.7709e-16**

Наименьший показатель степени двойки = 8

Относительная погрешность решения, полученного методом Чебышева = 3.29051e-16

График среднеквадратической нормы погрешности решения:


Среднеквадратическая норма погрешности
100.0
10.0
1.0
0.1
0.01
0.001
0.0001
1e-05
1e-06
1e-07
1e-08
1e-09
1e-10
1e-11
1e-12
1e-13
1e-14
1e-15
1e-16
1e-17
0
16
32
48
64
80
96
112
128
144
160
176
192
208
224
240
256
Номер итерации